# A Proposition for Sequence Mining Using Pattern Structures

Victor Codocedo[1,3] ✉, Guillaume Bosc[2], Mehdi Kaytoue[2], Jean-François Boulicaut[2], and Amedeo Napoli[3]

[1] Inria Chile,
firstname.secondname@inria.cl
[2] Université de Lyon. CNRS, INSA-Lyon, LIRIS,
firstname.secondname@insa-lyon.fr
[3] LORIA (CNRS – INRIA Nancy Grand-Est – Université de Lorraine),
firstname.secondname@loria.fr

**Abstract.** In this article we present a novel approach to rare sequence mining using pattern structures. Particularly, we are interested in mining closed sequences, a type of maximal sub-element which allows providing a succinct description of the patterns in a sequence database. We present and describe a sequence pattern structure model in which rare closed subsequences can be easily encoded. We also propose a discussion and characterization of the search space of closed sequences and, through the notion of sequence alignments, provide an intuitive implementation of a similarity operator for the sequence pattern structure based on directed acyclic graphs. Finally, we provide an experimental evaluation of our approach in comparison with state-of-the-art closed sequence mining algorithms showing that our approach can largely outperform them when dealing with large regions of the search space.

## 1 Introduction

Sequence mining is an interesting application of data analysis through which we aim at finding patterns in strings of symbols, sets or events [1]. One of the simplest incarnations of this problem is finding, within a database of words, a set of substrings that appear more frequently among them, e.g. prefixes or suffixes are usually "frequent substrings".

Traditional algorithms for sequence mining rely on prefix enumeration [11–13] exploiting the fact that the longest a prefix is, the fewer words contain it (e.g. prefix pr- is contained in prehistoric, prehispanic and primitive, while the prefix pri- is only contained in primitive).

While these techniques are usually very efficient at finding frequent subsequences, there are some issues they do not address. Firstly, frequent sequences are usually short and very simple in structure. Because of this, current sequence miners are not able to find patterns with certain complexity restrictions such as minimal length. Secondly, prefix enumeration techniques usually provide numerous sequences as a result of the mining process. In this regard, we may be

interested in a subset of the results using notions of maximality (what are known as closed subsequences) which provide a succinct representation of the patterns in a database (CloSpan, ClaSP or BIDE [12, 11, 7]).

In this article we present a novel approach for rare subsequence mining using the FCA framework. We adapt the pattern structures extension to deal with complex object descriptions [6] to build a mining tool that naturally models "closed sequences". By defining a similarity operator between sequence sets we are able to derive a *rare* sequence mining tool that is able to retrieve large subsequences within a dataset.

The remainder of this article is organized as follows. Section 2 introduces the main notions behind the problem of sequence mining. Section 3 models the problem in the framework of pattern structures. Section 4 formalizes the implementation of our mining technique. Section 5 presents a discussion on the state-of-the-art algorithms for sequence mining. Section 6 presents the experimental evidence to support our findings and a discussion about their implications. Finally, Section 7 concludes the article with a summary of our work.

## 2 Formalization

Before introducing our contributions, let us provide some basic definitions of the sequence mining problem. Let $\mathtt{M}$ be a set of items or symbols, a sequence of itemsets is an ordered set denoted as $< \alpha_1, \alpha_2, \alpha_3..., \alpha_n >$ where $\alpha_i \subseteq \mathtt{M}$, $i \in [1, n]$. Alternatively, we will use a *bar notation* $\alpha_1|\alpha_2|\alpha_3|...|\alpha_n$. A sequence with ID (SID) $\alpha$ is denoted as $\alpha := \alpha_1|\alpha_2|\alpha_3|...|\alpha_n$ and is referred as well as sequence $\alpha$. The *size* of sequence $\alpha$ is denoted as $size(\alpha) = n$ while its length is denoted as $len(\alpha) = \sum |\alpha_i|$ with $i \in [1, n]$ where $|\cdot|$ indicates set cardinality. A set of sequences $\mathcal{A}$ is called a *sequence database*. For the sake of simplicity we will drop the set parentheses for the itemsets within a sequence using the bar notation, e.g. $\{a, b\}|\{c, d\}$ is denoted as $ab|cd$.

**Definition 1** SUBSEQUENCE. *Sequence $\beta := \beta_1|\beta_2|....|\beta_m$ is a subsequence of sequence $\alpha := \alpha_1|\alpha_2|...|\alpha_n$ (denoted as $\beta \preceq \alpha$) iff $m \leq n$ and if there exists a a sequence of natural numbers $i_1 < i_2 < i_3 < ... < i_m$ such that $\beta_j \subseteq \alpha_{i_j}$ with $j \in [1, m]$ and $i_j \in [1, n]$. We denote as $\beta \prec \alpha$ when $\beta \preceq \alpha$ and $\beta \neq \alpha$.*

For example, consider sequence $\beta := a|b|a$ and sequence with SID $\alpha^2$ in Table 1. In this particular case, there exists a sequence of numbers $\langle 1, 3, 4\rangle$ where we have: The first itemset of $\beta$ is a subset of the *first* itemset of $\alpha^2$ ($\{a\} \subseteq \{a, d\}$). The second itemset of $\beta$ is a subset of the *third* itemset of $\alpha^2$ ($\{b\} \subseteq \{b, c\}$). The third itemset of $\beta$ is a subset of the *fourth* itemset of $\alpha^2$ ($\{a\} \subseteq \{a, e\}$). We conclude that $\beta \prec \alpha^2$ (since $\beta \neq \alpha^2$). Given a sequence database $\mathcal{A}$ we will consider the set of all sub-sequences of elements in $\mathcal{A}$ denoted as $\mathcal{S}$ where $\beta \in \mathcal{S} \iff \exists \alpha \in \mathcal{A}$ s.t. $\beta \preceq \alpha$ ($\mathcal{A} \subseteq \mathcal{S}$).

**Definition 2** SUPPORT OF A SEQUENCE. *Let $\beta \in \mathcal{S}$ be a sequence. The support of $\beta$ w.r.t. $\mathcal{A}$ is defined as: $\sigma(\beta) = |\{\alpha \in \mathcal{A} \mid \beta \preceq \alpha\}|$*

With $\beta := a|b|a$ we have that $\beta \prec \alpha^1, \alpha^2$ and thus $\sigma(\beta) = 2$.

**Definition 3** Closed Sequence. *$\beta^1 \in \mathcal{S}$ is a closed sequence w.r.t $\mathcal{A}$ iff $\nexists\ \beta^2 \in \mathcal{S}$ such that $\beta^1 \prec \beta^2$ and $\sigma(\beta^1) = \sigma(\beta^2)$.*

Consider sequences $\beta^1 := a|b|a$ and $\beta^2 := a|bc|a$ as subsequences of elements in $\mathcal{A}$ of Table 1. It is easy to show that $\sigma(\beta^2) = 2$. Given that $\beta^1 \prec \beta^2$ and that they have the same support, we conclude that $\beta^2$ is not a closed sequence w.r.t. $\mathcal{A} = \{\alpha^1, \alpha^2, \alpha^3, \alpha^4\}$.

When mining subsequences, those that are closed in the sense of Definition 3 provide a succinct result, critically compacter than the set of all possible subsequences [12, 11]. Consequently, in what follows we will be interested only in sets of closed sequences.

## 3 The Pattern Structure of Sequences.

The pair $(\mathcal{S}, \preceq)$ composed of a sequence set and a subsequence order constitutes a partially ordered set. However, it does not conform a proper lattice structure, since in general, two sequences have more than one common subsequence (the infimum of two sequences is not unique).

In the context of sequence mining, this fact has been already noticed in [13], where the space of sequences and their order is denominated a *hyper-lattice*. In the context of FCA, this problem is analogous to the one introduced in [6] for graph pattern mining. Indeed, we can properly embed the *hyper-lattice* of partially ordered sequences into a lattice of sequence sets using the framework of pattern structures.

**Definition 4** Set of Closed Sequences. *Let $\mathtt{d} \subseteq \mathcal{S}$ be a set of sequences, we call $\mathtt{d}^+ \subseteq \mathtt{d}$ a set of closed sequences iff $\mathtt{d}^+ = \{\beta^i \in \mathtt{d} \mid \nexists \beta^j \in \mathtt{d}$ s.t. $\beta^i \prec \beta^j\}$.*

The intuition behind Definition 4 is that given a set of sequences $\mathtt{d}$, we can consider every sequence inside it as having the same support. Thus, $\mathtt{d}^+$ contains *closed sequences* in the sense of Definition 3.

**Definition 5** Set of Common Closed Subsequences (SCCS). *Consider two sequences $\beta^i, \beta^j \in \mathcal{S}$, we define their meet $\wedge$ as follows:*

$$\beta^i \wedge \beta^j = \{\beta \in \mathcal{S} \mid \beta \preceq \beta^i \text{ and } \beta \preceq \beta^j\}^+$$

Intuitively, $\beta^i \wedge \beta^j$ corresponds to a set of common subsequences to $\beta^i$ and $\beta^j$. As indicated by the notation, we will require this set to contain only closed sequences (Definition 4) and thus we will denote it as the set of common closed subsequences (SCCS) of $\beta^i$ and $\beta^j$.

**Definition 6** SCCS for sequence sets. *Given two sets of closed sequences $\mathtt{d}^1, \mathtt{d}^2 \subseteq \mathcal{S}$, the similarity operator between them is defined as follows:*

$$\mathtt{d}^1 \sqcap \mathtt{d}^2 = \left\{ \bigcup_{\substack{\beta^i \in \mathtt{d}^1 \\ \beta^j \in \mathtt{d}^2}} \beta^i \wedge \beta^j \right\}^+$$

In a nutshell, $\sqcap$ is the SCCS between the elements within two sets of closed sequences. Let $\mathtt{D} = \{\mathtt{d} \subseteq \mathcal{S} \mid \mathtt{d} = \mathtt{d}^+\}$ denominate the *set of all sets of closed sequences in* $\mathcal{S}$ , then we have $\wedge : \mathcal{S}^2 \rightarrow \mathtt{D}$ and $\sqcap : \mathtt{D}^2 \rightarrow \mathtt{D}$.

For a sequence $\beta \in \mathcal{S}$, $\delta(\beta) = \{\beta^i \in \mathcal{S} \mid \beta^i \preceq \beta\}^+$ denotes its set of closed subsequences. Trivially, the set of closed subsequences of a single sequence $\beta$ contains a single element which is itself, i.e. $\delta(\beta) = \{\beta\}$. From this, it follows that given any three sequences $\beta^i, \beta^j, \beta^k \in \mathcal{S}$ we have that $\beta^i \wedge \beta^j = \delta(\beta^i) \sqcap \delta(\beta^j)$ and $(\delta(\beta^i) \sqcap \delta(\beta^j)) \sqcap \delta(\beta^k)$ properly represents the SCCS of $\beta^i, \beta^j$ and $\beta^k$. For example, consider sequences with $\alpha^1$ and $\alpha^3$ in Table 1.

$$\delta(\alpha^1) \sqcap \delta(\alpha^3) = \{a|abc|ac|d|cf\} \sqcap \{ef|ab|df|c|b\} = \{ab|d|c, ab|f, a|b\}$$

Clearly, $\sqcap$ is idempotent and commutative as it inherits these properties from $\wedge$. Furthermore, it can be easily shown that $\sqcap$ is also associative. Using these properties, we can provide a more complex example. Consider $\delta(\alpha^1) \sqcap \delta(\alpha^2) \sqcap \delta(\alpha^3) \sqcap \delta(\alpha^4)$ from Table 1. Since we already have the result for $\delta(\alpha^1) \sqcap \delta(\alpha^3)$, we re-arrange and proceed as follows:

$$\begin{aligned} \delta(\alpha^1) \sqcap \delta(\alpha^2) \sqcap \delta(\alpha^3) \sqcap \delta(\alpha^4) &= (\delta(\alpha^1) \sqcap \delta(\alpha^3)) \sqcap (\delta(\alpha^2) \sqcap \delta(\alpha^4)) \\ \delta(\alpha^1) \sqcap \delta(\alpha^3) &= \{ab|d|c, ab|f, a|b\} \\ \delta(\alpha^2) \sqcap \delta(\alpha^4) &= \{a|c|b, a|c|c, e\} \end{aligned}$$

Now, we need to calculate $\{ab|d|c, ab|f, a|b\} \sqcap \{a|c|b, a|c|c, e\}$ which requires 9 SCCS calculations. We show those with non-empty results.

$$\begin{array}{ccc} ab|d|c \wedge a|c|b = \{a|c, b\} & ab|d|c \wedge a|c|c = \{a|c\} & ab|f \wedge a|c|b = \{a, b\} \\ ab|f \wedge a|c|c = \{a\} & a|b \wedge a|c|b = \{a|b\} & a|b \wedge a|c|c = \{a\} \end{array}$$

The union of these results gives us $\{a, b, a|b, a|c\}$ from which we remove $a$ and $b$ which are not closed subsequences in the set. Finally, we have:

$$\delta(\alpha^1) \sqcap \delta(\alpha^2) \sqcap \delta(\alpha^3) \sqcap \delta(\alpha^4) = \{a|b, a|c\}$$

Indicating that the set of closed subsequences common to all sequences in Table 1 only contains $a|b$ and $a|c$.

**Definition 7** Order in D. *Let* $\mathtt{d}^1, \mathtt{d}^2 \in \mathtt{D}$*, we say that* $\mathtt{d}_1$ *is subsumed by* $\mathtt{d}_2$ *(denoted as* $\mathtt{d}_1 \sqsubseteq \mathtt{d}_2$*) iff:*

$$\begin{aligned} \mathtt{d}_1 \sqsubseteq \mathtt{d}_2 &\iff \forall \beta^i \in \mathtt{d}_1 \; \exists \beta^j \in \mathtt{d}_2 \text{ s.t. } \beta^i \preceq \beta^j \\ \mathtt{d}_1 \sqsubset \mathtt{d}_2 &\iff \mathtt{d}_1 \sqsubseteq \mathtt{d}_2 \text{ and } \mathtt{d}_1 \neq \mathtt{d}_2 \end{aligned}$$

**Proposition 1** Semi-lattice of sequence descriptions. *Let* D *be the set of all sets of closed sequences (hereafter indistinctly referred to as "the set of sequence descriptions") built from itemset* M*. The pair* $(\mathtt{D}, \sqcap)$ *is a semi-lattice of sequence descriptions where* $\mathtt{d}_1 \sqsubseteq \mathtt{d}_2 \iff \mathtt{d}_1 \sqcap \mathtt{d}_2 = \mathtt{d}_1$ *for* $\mathtt{d}_1, \mathtt{d}_2 \in \mathtt{D}$.

The proof of $\mathtt{d}_1 \sqsubseteq \mathtt{d}_2 \iff \mathtt{d}_1 \sqcap \mathtt{d}_2 = \mathtt{d}_1$ in Proposition 1 is straightforward and follows from Definitions 6 and 7.

**Definition 8** THE PATTERN STRUCTURE OF SEQUENCES. *Let $\mathcal{A}$ be a sequence database, $(\mathtt{D}, \sqcap)$ the semi-lattice of sequence descriptions and $\delta : \mathcal{S} \rightarrow \mathtt{D}$ a mapping assigning to a given sequence its corresponding set of closed subsequences (recall that $\mathcal{A} \subseteq \mathcal{S}$). The pattern structure of sequences is defined as:*

$$\mathcal{K} = (\mathcal{A}, (\mathtt{D}, \sqcap), \delta)$$

For the sake of brevity we do not provide the development of the pattern structure framework which can be found in [6]. However, let us introduce some concepts and notation that will result important in the remainder of the article. The pair $(\mathtt{A}, \mathtt{d})$ with $\mathtt{A} \subseteq \mathcal{A}$ and $\mathtt{d} \in \mathtt{D}$ is a *sequence pattern concept* where $\mathtt{A}^{\square} = \mathtt{d}$ and $\mathtt{d}^{\square} = \mathtt{A}$. The derivation operator $(\cdot)^{\square}$ is defined dually for extents and pattern descriptions. The order of sequence pattern concepts is denoted as $(\mathtt{A}_1, \mathtt{d}_1) \leq_{\mathcal{K}} (\mathtt{A}_2, \mathtt{d}_2)$ iff $\mathtt{A}_1 \subseteq \mathtt{A}_2$ and $\mathtt{d}_2 \sqsubseteq \mathtt{d}_1$ where $\mathtt{A}_1, \mathtt{A}_2 \subseteq \mathcal{A}$ and $\mathtt{d}_1, \mathtt{d}_2 \in \mathtt{D}$. Finally, the set of all sequence pattern concepts with the order $\leq_{\mathcal{K}}$ defines a sequence pattern concept lattice denoted as $(\mathfrak{B}(\mathcal{K}), \leq_{\mathcal{K}})$.

Within a sequence pattern concept we have that $\mathtt{d}^{\square\square} = \mathtt{d}$, i.e. the set of sequences $\mathtt{d}$ is closed. This should not be confused with the notion of *set of closed sequences* given in Definition 4 which is a property of the elements inside $\mathtt{d}$. Actually, $\mathtt{d}$ is a set of common closed subsequences (SCCS) when $\mathtt{d}^{+} = \mathtt{d}$. With this new constraint, the correct denomination for $\mathtt{d}$ would be a *closed set of common closed subsequences*. To avoid confusions, hereafter we will denominate a set $\mathtt{d}$ such that $\mathtt{d}^{+} = \mathtt{d}$ and $\mathtt{d}^{\square\square} = \mathtt{d}$ as a *maximal SCCS*.

A sequence pattern concept represents a set of sequences $\mathtt{A}$ in the database and their maximal SCCS, i.e. closed subsequences of elements in $\mathtt{A}$ are contained in $\mathtt{d}$. Consider from the previous example that $\{e|a|c|b\}^{\square} = \{\alpha^3, \alpha^4\}$. Now, we can calculate $\{\alpha^3, \alpha^4\}^{\square} = \{e|a|c|b, e|f|c|b, e|b|c, f|b|c\}$. From this it follows that $\{e|a|c|b\}$ is not a maximal SCCS and that $(\{\alpha^3, \alpha^4\}, \{e|a|c|b, e|f|c|b, e|b|c, f|b|c\})$ is a sequence pattern concept. Figure 1 shows the sequence pattern concept lattice built from entries in Table 1. The bottom concept in the figure is a *fake* element in the lattice since there is no definition for the meet between pattern concepts. We have included it to show that it represents the concept with empty extent, i.e. the "subsequences of no sequences". The indefinition of the meet between formal concepts also means that we can only *join* sequence pattern concepts starting from object concepts $\gamma(\alpha) = (\{\alpha\}, \delta(\alpha))$. Consequently our mining approach is a *rare* sequence mining technique.

The concept lattice structure allows proving a characterization of an important property of sequences, namely we can show that all sequences in an intent are closed w.r.t. the database $\mathcal{A}$.

**Proposition 2** *Given a sequence pattern concept $(\mathtt{A}, \mathtt{d})$ any sequence $\beta \in \mathtt{d}$ is not only closed w.r.t. $\mathtt{A}$ but it is also closed w.r.t. $\mathcal{A}$.*

*Proof.* Consider two sequence pattern concepts $(\mathtt{A}_1, \mathtt{d}_1), (\mathtt{A}_2, \mathtt{d}_2) \in \mathfrak{B}(\mathcal{K})$ such as $(\mathtt{A}_1, \mathtt{d}_1) <_{\mathcal{K}} (\mathtt{A}_2, \mathtt{d}_2)$ (then $\mathtt{A}_1 \subset \mathtt{A}_2$ and $\mathtt{d}_2 \sqsubset \mathtt{d}_1$). Particularly, the latter relation indicates that $\forall \beta^j \in \mathtt{d}_2\ \exists \beta^i \in \mathtt{d}_1$ s.t. $\beta^j \preceq \beta^i$ and $\mathtt{d}_1 \neq \mathtt{d}_2$ (Definition 7). In the case that $\beta^j \prec \beta^i$, the fact that $\mathtt{A}_1 \subset \mathtt{A}_2 \implies |\mathtt{A}_1| < |\mathtt{A}_2|$ secures that both sequences have different supports making $\beta^j$ closed. In the case that $\beta^j = \beta^i$,

there are two different cases. Either a third concept $(\mathtt{A}_3, \mathtt{d}_3) \in \mathfrak{B}(\mathcal{K})$ exists such that $(\mathtt{A}_2, \mathtt{d}_2) <_{\mathcal{K}} (\mathtt{A}_3, \mathtt{d}_3)$ where $\beta^j \in \mathtt{d}_2$ and $\beta^j \notin \mathtt{d}_3$ and thus, we fall in the previous case and $\beta^j$ is closed, or $\beta^j \in \mathcal{A}^{\square}$ and $\beta^j$ is closed with support $|\mathcal{A}|$.

Indeed, since $(\mathtt{A}^{\square}) = (\mathtt{A}^{\square})^+$, any $\beta \in \mathtt{A}^{\square}$ is closed w.r.t. $\mathcal{A}$, even *when we may not know the support for that particular sequence*. Moreover, the support of any closed sequence $\beta$ is given by the cardinality of the extent of the concept $(\mathtt{A}_1, \mathtt{d}_1)$ where $\beta \in \mathtt{d}_1$ s.t. $\nexists(\mathtt{A}_2, \mathtt{d}_2)$ where $(\mathtt{A}_1, \mathtt{d}_1) <_{\mathcal{K}} (\mathtt{A}_2, \mathtt{d}_2)$ and $\beta \in \mathtt{d}_2$.

For example, consider Figure 1 and the concept labelled as $\{\alpha^1, \alpha^2\}$ containing sequence $a|c|c$. While we can be sure that $a|c|c$ is a closed subsequence w.r.t. $\mathcal{A}$, the support is not 2. Actually, $a|c|c$ exists also in the intent of the concept labelled $\{\alpha^1, \alpha^2, \alpha^4\}$. Since it does not exists in the suprema (the only superconcept of the latter), we can conclude that the support of $a|c|c$ is 3.

Finally, we briefly mention the pattern complexity filtering capabilities of our approach. For a relation $(\mathtt{A}_1, \mathtt{d}_1) <_{\mathcal{K}} (\mathtt{A}_2, \mathtt{d}_2)$, sequences in $\mathtt{d}_2$ will be shorter and with fewer items per itemset than those in $\mathtt{d}_1$. Given thresholds for size and lenght, the sequence search space can be pruned similarly to support pruning .

## 4 Implementing the similarity operator $\sqcap$

So far we have taken for granted the ability to calculate the set of common closed subsequences (SCCS) of two sequence sets (i.e. $\mathtt{d}_1 \sqcap \mathtt{d}_2$). In what follows, we will describe and discuss our approach to achieve this.

### 4.1 Rationale

Sequences are composed of two parts, firstly the elements in the sequence and secondly, the order they have. Considering the latter, we can think about how a "common subsequence" (an element in the SCCS) $\beta$ of two sequences $\beta^1, \beta^2$ relates to them. Given the definition of subsequence (Definition 1), since $\beta \preceq \beta^1$

**Table 1.** An example sequence database.

| *sid* | Sequence |
|---|---|
| $\alpha^1$ | $a\|abc\|ac\|d\|cf$ |
| $\alpha^2$ | $ad\|c\|bc\|ae$ |
| $\alpha^3$ | $ef\|ab\|df\|c\|b$ |
| $\alpha^4$ | $e\|g\|af\|c\|b\|c$ |

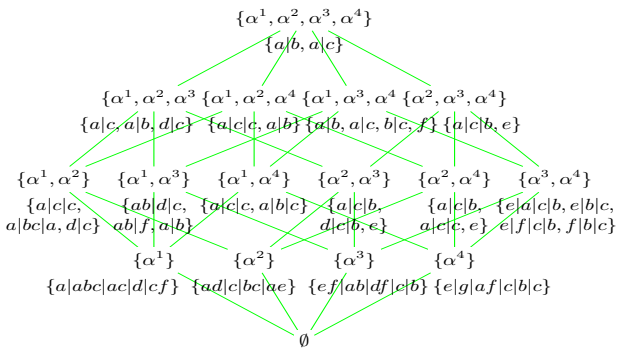



**Fig. 1.** Extents & Intents of the sequence pattern structure built from Table 1.

we have a sequence of integers $i_1^1 < i_2^1 < i_3^1 < ... < i_m^1$ such that $\beta_j \subseteq \beta_{i_j}^1$ with $i_j^1 \in [1, n]$ and $j \in [1, m]$ with $size(\beta) = m$, $size(\beta^1) = n$, and $m < n$. A similar sequence of integers can be derived from the relation $\beta \preceq \beta^2$, namely $i_1^2 < i_2^2 < i_3^2 < ... < i_m^2$. Since both sequence of integers have the same length, we can arrange them in a sequence of tuples $P = \langle (i_1^1, i_1^2), (i_2^1, i_2^2), \ldots, (i_m^1, i_m^2) \rangle$. Furthermore, given that $\beta_j \subseteq \beta_{i_j}^1$ and $\beta_j \subseteq \beta_{k_j}^2$, it is only natural to define $\beta_j = \beta_{i_j}^1 \cap \beta_{k_j}^2$ (notice that defining $\beta_j \subset \beta_{i_j}^1 \cap \beta_{k_j}^2$ would render $\beta$ not a closed sequence). Consequently, tuple $P$ is a possible characterization of $\beta$ in terms of its parents supersequences $\beta^1$ and $\beta^2$. Actually, given any set of sequences $\mathtt{A}$, a common subsequence of them $\beta \in \mathtt{A}^{\square}$ has one or more characterizations $P$.

Generally, we call $P := \langle t_1, t_2, \ldots, t_m \rangle$ an alignment with size $m$ of sequences in $\mathtt{A}$ where $r = |\mathtt{A}|$, $n_k$ is the size of the $k$-th sequence in $\mathtt{A}$ with $k \in [1, r]$, $t_j = (i_j^k)$, and $j \in [1, m]$ ($t_j \in \{1, 2, \ldots, n\}^r$). Furthermore, we will require that with $k \in [1, r]$ and $j \in [1, (m-1)]$ we have that: (i) $(\exists i_1^k \in t_1)$ s.t. $i_1^k = 1$, (ii) $(\exists i_m^k \in t_m)$ s.t. $i_m^k = n_k$, (iii)$(\exists i_{j+1}^k \in t_{j+1})$ s.t. $i_{j+1}^k = i_j^k + 1$, and (iv) $(\forall i_{j+1}^k \in t_{j+1})$ $i_j^k < i_{j+1}^k$. Conditions (i), (ii) and (iii) secure that alignments always begin with the first element of at least one sequence in $\mathtt{A}$, always end with the last element of a least one sequence in $\mathtt{A}$, and that they are *maximal*, e.g. $\langle (1,1), (3,3) \rangle$ is not a maximal alignment if $\langle (1,1), (2,2), (3,3) \rangle$ exists. Condition (iv) secures that alignments only consider incremental tuples, e.g. $\langle (1,2), (2,1) \rangle$ is not an alignment. $\mathcal{P}_n^r$ is the space of all maximal alignments of sequences in $\mathtt{A}$.

It is possible to show that $|\mathtt{A}^{\square}| \leq |\mathcal{P}_n^r|$ provided that conditions (i), (ii), (iii) and (iv) hold for alignments in $\mathcal{P}_n^r$ and that $|\mathtt{M}| = n^r$. This is demonstrated by showing that there exists a scenario (however unlikely) where for each different closed subsequence there exists a unique alignment, i.e. $(\forall \beta \in \mathtt{A}^{\square})(\exists! \langle t_1, t_2, \ldots, t_m \rangle \in \mathcal{P}_n^r)$ s.t. $\beta_j = \bigcap \alpha_{t_j^i}^i = \pi(t_j)$ where $\pi : \mathbb{N}^r \to \mathbb{N}$ is pairing function encoding tuples in $\{1, 2, \ldots, n\}^r$ into $\mathtt{M} = \{\pi(t_j) \mid t_j \in \{1, 2, \ldots, n\}^r\}$. Thus, given a set of sequences $\mathtt{A}$ we only need to compute *all possible alignments* in order to obtain their *complete set of common closed subsequences* $\mathtt{A}^{\square}$ (SCCS).

Notably, this relation allows calculating the number of possible closed subsequences between two sequences of arbitrary length[4]. Table 2 shows this number with $n \in [1, 10]$. For example, for two sequences of size 10 we have a maximum of 26797 alignments/closed subsequences (requiring $|\mathtt{M}| = 2^{10} = 1024$ items).

Enumerating all possible alignments may look an extremely naive strategy considering how the space of alignment grows w.r.t. the number of sequences in $\mathtt{A}$ and their size, however the way we encode alignments makes the representation of subsequences more compact than simply listing them. In addition, the sparsity of sequence element intersections (most $\beta_j = \beta_{i_j}^1 \cap \beta_{k_j}^2$ are empty) provides a very efficient pruning method. In the following, we show how to encode alignments in $\mathcal{P}_n^r$ as paths of a directed acyclic graph (DAG).

**Definition 9** DAG OF ALIGNMENTS. *We call $\mathcal{G}_n^r = (V, E, \lambda)$ a directed acyclic graph of alignments $\mathcal{P}_n^r$ (DAG of alignments) for a set of $r$ sequences $\mathtt{A} \subseteq \mathcal{A}$ of*

[4] `https://oeis.org/A171155`

*size $n$, if the set of vertices,edges and labelling function $\lambda : V \to \wp(\mathtt{M})$ are:*

$$V = \{t_j = \langle i_j^k \rangle \mid t_j \in \{1, 2, \ldots, n\}^r\} \cup \{0_r, (n+1)_r\}$$
$$E = \{(t_a, t_b) \iff (\exists i_b^k \in t_b)\, i_b^k = i_a^k + 1 \text{ and } (\forall i_b^k \in t_b)\, i_a^k < i_b^k\}$$
$$\lambda(t_j) = \bigcap_{i \in [1,r]} \alpha_{i_j^i}^i$$

*$0_r, (n+1)_r$ are $r$-tuples filled with 0 and $n+1$ values respectively. We denominate $0_r$ the source vertex (or source node) and $(n+1)_r$ the sink vertex (or sink node).*

Notice that conditions (iii) and (iv) for alignments in $\mathcal{P}_n^r$ described above are encoded in the definition of edges $E$ in $\mathcal{G}_n^r$. The inclusion of the source and sink nodes ensure that all paths between them corresponds to alignments in $\mathcal{P}_n^r$ that also respect conditions (i) and (ii). Thus, the DAG $\mathcal{G}_n^r$ is designed so that all possible maximal alignments among $r$ sequences of size $n$ correspond to *paths between the source and sink nodes*. In addition, all possible common subsequences can be read from the graph by means of the labelling function $\lambda$. Trivially, for a set of sequences with different sizes, a DAG of alignments can be built with $n$ corresponding to the maximal sequence size in the set. Figure 2 shows the DAG $\mathcal{G}_4^2$ without labels, where the number of paths between $(0, 0)$ and $(5, 5)$ is 27 (as predicted by Table 2).

### 4.2 Calculating $\delta(\alpha^1) \sqcap \delta(\alpha^2)$

Consider sequences $\alpha^1 = ab|cd|e|f$, $\alpha^2 = b|ac|d|ef$ and their itemset intersections shown in Table 3 (empty cells indicate an empty intersection) Since both sequences have size 4, we can use the DAG $\mathcal{G}_4^2$ shown in Figure 2 to obtain all possible alignments between these sequences. Then, we can derive associated subsequences using the labelling function given by Table 3. In these subsequences the SCCS must be included.

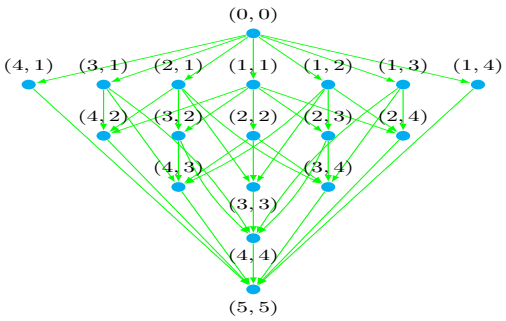



**Fig. 2.** DAG for the alignments of two sequences of size 4

**Table 2.** Number of maximal alignments for $n \in [1, 10]$.

| $n$ | $\lvert\mathcal{P}_n^2\rvert$ |
|---|---|
| 1 | 1 |
| 2 | 3 |
| 3 | 9 |
| 4 | 27 |
| 5 | 83 |
| 6 | 259 |
| 7 | 817 |
| 8 | 2599 |
| 9 | 8323 |
| 10 | 26797 |

In this case more than half the itemset intersections are empty which makes the task of deriving paths easier. Figure 3 shows a subgraph of $\mathcal{G}_4^2$, where vertex $(i,j)$ is labelled as $\lambda((i,j)) = \alpha_i^1 \sqcap \alpha_j^2$. Vertices $v \in V$ for which $\lambda(v) = \emptyset$ were removed from Figure 3. We have also connected $(2,2)$ with $(4,4)$ since vertex $(3,3)$ is among those vertices removed.

A path between $(0,0)$ and $(5,5)$ is $\langle(1,1),(2,2),(4,4)\rangle$ (removing the source and sink nodes). Thus, a subsequence can be derived using the labels of the vertices in the path, i.e. $\beta = \lambda((1,1))|\lambda((2,2))|\lambda((4,4))$ or $\beta = b|c|f$. All the subsequences of $\alpha^1$ and $\alpha^2$ are shown in Table 4 from which we can derive that the SCCS of $\alpha^1$ and $\alpha^2$ is $\delta(\alpha^1) \sqcap \delta(\alpha^2) = \{b|c|e, b|c|f, b|d|e, b|d|f, a|d|e, a|d|f\}$.

This result illustrates the usefulness of the DAG of alignments in Figure 2. Indeed, instead of enumerating all possible sequences (like in usual subsequence mining algorithms) we just need to intersect the itemsets of two sequences and later interpret them using the corresponding DAG.

To extend the example, let us consider a third sequence $\alpha^3 = ab|c|d|ef$ and the SCCS of $\alpha^1, \alpha^2$ and $\alpha^3$. We need to calculate the table of intersections that in this case corresponds to a *cube of intersections* where the cell $(i,j,k)$ in the cube contains $\alpha_i^1 \sqcap \alpha_j^2 \sqcap \alpha_k^3$. While the construction of this cube requires $4^3 = 64$ intersections, using the results in Table 4 we need only to calculate 24 intersections (those with empty intersections in Table 4 can be disregarded). Using the DAG of alignments $\mathcal{G}_4^3$, we can obtain the SCCS such that $\delta(\alpha^1) \sqcap \delta(\alpha^2) \sqcap \delta(\alpha^3) = \{b|c|e, b|c|f, b|d|e, b|d|f, a|d|e, a|d|f\}$.

### 4.3 Discussion

In the previous example, one important thing to notice is that we did not need to interpret the sequences from the intersection table of $\mathtt{d} = \{\alpha^1, \alpha^2\}^{\square}$ in order to calculate $\mathtt{d} \sqcap \delta(\alpha^3)$. This information is encoded in the table itself in the form of

**Table 3.** Intersection table for sequences $\alpha^1 = ab|cd|e|f$ and $\alpha^2 = b|ac|d|ef$.

| | $\alpha_1^2$ | $\alpha_2^2$ | $\alpha_3^2$ | $\alpha_4^2$ |
|---|---|---|---|---|
| $\alpha_1^1$ | $b$ | $a$ | | |
| $\alpha_2^1$ | | $c$ | $d$ | |
| $\alpha_3^1$ | | | | $e$ |
| $\alpha_4^1$ | | | | $f$ |

**Table 4.** List of paths in the DAG of Figure 3 and subsequences derived.

| Path | Sequence |
|---|---|
| $\langle(1,1),(2,2),(4,4)\rangle$ | $b\|c\|f$ |
| $\langle(1,1),(2,2),(3,4)\rangle$ | $b\|c\|e$ |
| $\langle(1,1),(2,3),(4,4)\rangle$ | $b\|d\|f$ |
| $\langle(1,1),(2,3),(3,4)\rangle$ | $b\|d\|e$ |
| $\langle(1,2),(2,3),(3,4)\rangle$ | $a\|d\|e$ |
| $\langle(1,2),(2,3),(4,4)\rangle$ | $a\|d\|f$ |

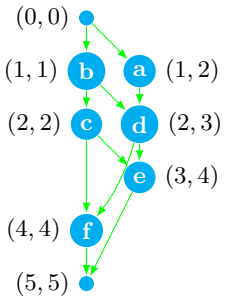



**Fig. 3.** DAG for sequences $\alpha^1 = ab|cd|e|f, \alpha^2 = b|ac|d|ef$.

index tuples of the itemsets we had intersected and can be recovered at any time using the corresponding DAG. Nevertheless, we are not spared from interpreting the sequences out of the table of intersections. This is because we still need the sequences to establish the order between two sequence sets. We have not found a property that would allow us to compare sequence sets from their intersection tables or from their DAG of alignments. Actually, this may not be possible. In this work we use the DAG of alignments just as a tool to explore and store the space of common closed subsequences. Implementation details of the DAG (such as the canonical order in the node tuples) were left out of the scope of this article for the sake of brevity, as well as some other elements of the search space exploration, e.g. the DAG allows filtering out sequences below a threshold of size and length.

## 5 Related Work

Sequence mining is usually based on what is called "prefix enumeration" or "pattern growth". This is, given a vocabulary `M` and a lexicographical order in it, we proceed by enumerating prefixes (using two main operations, namely *i-extension* and *s-extension* of sequences) and counting in a sequence database $\mathcal{A}$ how many elements contain them. The seminal paper in sequence mining is [1] introduces an Apriori-based algorithm to cut down the search space in a similar way to how it is done for standard itemset mining. A very popular extension of this idea was implemented in the PrefixSpan algorithm [10] using the notion of "projected databases" to decrease the number of comparisons in the database of sequences to calculate pattern support. PrefixSpan inspired different extensions such as CloSpan [6] to mine closed subsequences, OrderSpan [5] to mine *partially-ordered subsequences*, and CCSpan [14] to mine subsequences with a restriction of "contiguity" in the items they contain.

A different approach was taken in [13] using a "vertical database format" in an algorithm called SPADE (PrefixSpan-based algorithms are considered based on a "horizontal database format"). A rather interesting element of [13] is the characterization of the sequence search space as a *hyperlattice* which can be factored allowing parallel computation of patterns. A similar approach was adopted in BIDE [11] and ClaSP [7] for closed subsequence mining based on a vertical database format.

Partially-ordered subsequences is an idea introduced in [4] based on the notion that the subsumption order among sequences allows integrating different subsequences into a single pattern by means of a directed acyclic graph, which may provide a richer representation to an end user. This is closely related to our own model for sequence mining. Indeed, our own representation of sequence sets is through the DAG of alignments which are themselves partially-ordered patterns. This work inspired Frecpo [10] and OrderSpan [5] for mining closed partially-ordered subsequence patterns.

A comprehensive review on the sequence pattern mining techniques can be found in [9]. To the author's knowledge, our work is the first attempt to ap-

proach the problem of sequence mining in a general unrestricted manner using the framework of formal concept analysis. In this regard, an interesting work is presented in [2] where an FCA-based technique for mining Gradual patterns is introduced. Gradual patterns consider a total order between attribute values generating sequences of objects. The technique introduced allows characterizing sequences derived from these gradual patterns. However, much like BIDE, sequences are restricted to strings of elements, not sequences of itemsets. In [3], a different sequence mining framework based on pattern structures is introduced for dealing with medical records. These sequences contain complex elements composed of taxonomical elements and attribute sets. The authors decided to simplify the problem by considering subsequences containing just contiguous elements.

## 6 Experiments & Discussion

In this section, we present an experimental evaluation on our sequence mining approach over a series of synthetic datasets generated using the IBM Quest Dataset Generator software hosted in the SPMF Website[5]. Our approach was implemented in Python and embedded into a system named *Pypingen* available through a subversion repository [6].

### 6.1 Datasets and experimental setup

*Pypingen* was applied over 34 different datasets in 4 four groups. Each group allows showing how our algorithm handle different settings. The first group, denoted as $\mathcal{D}_{nseq_i}$, $i \in [1,7]$, contains datasets of different size $|\mathcal{A}|$, i.e. the number of sequences inside a dataset. The second group, $\mathcal{D}_{nitems_i}$, $i \in [1,12]$, contains datasets where the size of the itemset $|\mathtt{M}|$ varies. The third group, $\mathcal{D}_{size_i}$, $i \in [1,9]$, contains datasets of sequences that have different average sizes $mean(size)$, i.e. number of itemsets in a given sequence. Finally, the last group, $\mathcal{D}_{length_i}$, $i \in [1,6]$, consists of datasets where sequences have a varying average number of items by itemset, i.e. a variation in the *length* of sequences $mean(length/size)$ for a fixed size. Table 5 displays the different settings for each synthetic dataset generated. The variation in the parameters is a consequence of considering outputs that took *less than 15 minutes* to calculate. Experiments were performed on a 3.10 GHz processor with 8 GB main memory running Ubuntu 14.04.1 LTS.

### 6.2 Performance study - Comparison

All 34 datasets were also processed using "closed sequence pattern mining" algorithms ClaSP, CM-ClaSP, CloSpan and BIDE implemented in the SPMF soft-

[5] Open-source data mining library - `http://www.philippe-fournier-viger.com/spmf/`

[6] http://gforge.inria.fr/projects/pypingen

ware [7] (written in Java) under the same conditions of the execution of *Pypingen*. Algorithms were given a threshold value of 0 to obtain all closed sequence patterns. From all four algorithms, only CloSpan and BIDE were able to obtain results in less than 15 minutes. However, we do not report on these results for two reasons. Firstly, with respect to CloSpan, it was only able to obtain results for just a single dataset, namely $\mathcal{D}_{length_1}$ which is the simplest. Secondly, with respect to BIDE, even when it was able to obtain results in less time, these results were incorrect and incomplete. This is due to the fact that even when BIDE is catalogued as a "closed sequence miner", its formalization actually corresponds to that of a "closed string miner", that is, sequences where each itemset has cardinality 1. This simplification of the problem greatly reduces the search space and makes any comparison unfair. More importantly, in datasets where this condition is not met such as those in our evaluation, it produces incorrect (in the form of non-closed sequences) and incomplete results.

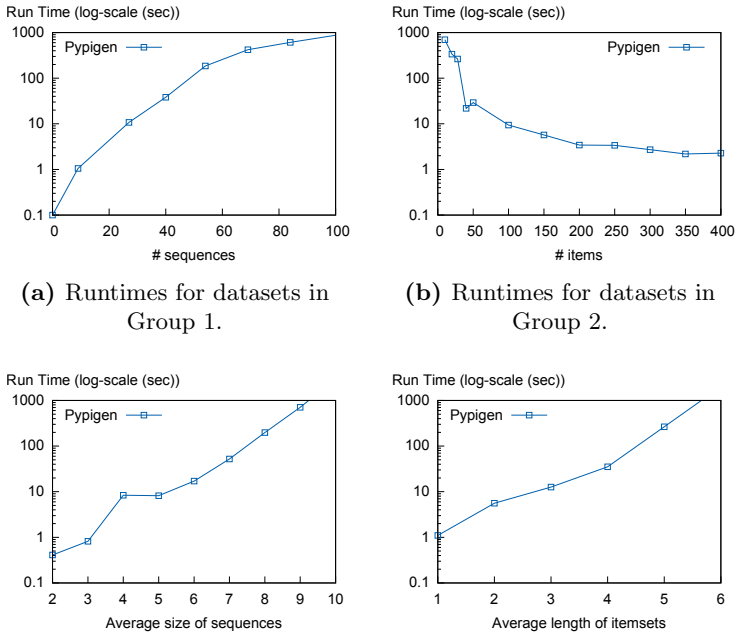
Execution times for *Pypingen* can be visualized in Figure 4 (execution times are given in logarithmic scale). The curves presented are consistent with the description of the algorithm. Runtimes when varying the size of the sequences (Figure 4c) grow exponentially due to the fact that the search space grows w.r.t. $n^r$ where $|\mathcal{A}| = r$ and $n$ is the maximum size of the sequences in the dataset. When varying the sequence length (Figure 4d), the search space does not change (it remains at $n^r$ which in this case is $5^{36}$). However, the larger the itemset cardinalities, the less empty intersections we can expect in the intersection table, and thus, the more closed subsequences for a sequence set.

Figure 4a is very interesting because it is more related to scalability. We can see that as the number of sequences grows, the time grows polynomially and not exponentially, which is desirable for a mining technique. Finally, Figure 4b

[7] Version 0.97d / 0.97e - 2015-12-06

**Table 5.** Characteristics of the datasets where $|\mathcal{A}|$ is the number of sequences in the dataset, $|\mathtt{M}|$ the cardinality of the set of items or symbols, $mean(size)$ the average size of the sequences of the dataset and $mean(length/size)$ the average number of items in the itemsets of the sequences.

| | |
|---|---|
| **Group 1** $\mathcal{D}_{nseq_i}$ | |
| $\lvert\mathtt{M}\rvert$ | 50 |
| $mean(size)$ | 5 |
| $mean(lenght/size)$ | 4 |
| $\lvert\mathcal{A}\rvert$ | $\{9, 27, 40, 54, 69, 84, 101\}$ |
| **Group 2** $\mathcal{D}_{nitems_i}$ | |
| $\lvert\mathtt{M}\rvert$ | $\{10, 20, 30, 40, 50, 100, 150, 200, 250, 300, 350, 400\}$ |
| $mean(size)$ | 5 |
| $mean(lenght/size)$ | 4 |
| $\lvert\mathcal{A}\rvert$ | 36 |
| **Group 3** $\mathcal{D}_{size_i}$ | |
| $\lvert\mathtt{M}\rvert$ | 50 |
| $mean(size)$ | $\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$ |
| $mean(lenght/size)$ | 4 |
| $\lvert\mathcal{A}\rvert$ | 21 |
| **Group 4** $\mathcal{D}_{lenght_i}$ | |
| $\lvert\mathtt{M}\rvert$ | 50 |
| $mean(size)$ | 5 |
| $mean(lenght/size)$ | $\{1, 2, 3, 4, 5, 6\}$ |
| $\lvert\mathcal{A}\rvert$ | 36 |




**(a)** Runtimes for datasets in Group 1.

**(b)** Runtimes for datasets in Group 2.

**(c)** Runtimes for datasets in Group 3.

**(d)** Runtimes for datasets in Group 4.

**Fig. 4.** Runtimes results for each dataset group.

shows that the execution time gets lower as the cardinality of itemset `M` gets larger. Indeed, since the remainder dataset parameters are left unchanged, the increase in the number of possible items leads to sparser datasets where most intersections are empty.

Figure 5 presents the relation between the number of closed subsequences and the execution time to calculate them using *Pypingen* over 1200 different synthetic datasets (each circle represents a dataset) created for 60 different configurations of vocabulary size (`M`), average size of sequences and average length of itemsets. All datasets contained only 10 sequences ($|\mathcal{A}| = 10$). Axes in the figure are provided in logarithmic scale, while the curve is the best fit to the data w.r.t. the coefficient of determination $R^2 = 0.97$ (a quadratic curve has $R^2 = 0.78$ while an exponential has $R^2 = 0.29$). We can appreciate that the curve is a power function with an exponent very close to 1 indicating a quasilinear relation between the number of closed sequences in a dataset and execution time required to calculate them, which is a desirable property for a pattern mining algorithm. Nevertheless, the performance of the algorithm slowly degrades as the number of patterns in a dataset increases. The radius of each point in the figure represents the ratio between the number of closed sequences and the number of sequence patterns found for that dataset.

### 6.3 Discussion

Algorithms CloSpan, ClaSP, CM-ClaSP and BIDE are *frequent* sequence miners, while *Pypingen* is a *rare* sequence miner. This means that the only "fair" comparison requires for all algorithms to explore the entire search space and mine all possible closed sequences. Under this circumstances, *Pypingen* clearly outperforms all of them as they are not able to achieve this task for all but one dataset in our experiments.

Arguably, our comparison can be understood as "fair" considering that all the algorithms are set to the worst case scenario. Frequent mining algorithms are designed under the assumption that results are in the "upper" part of the search space. Thus, asking them to explore the entire search space goes against the very assumption behind their designs and it becomes *unfair* to evaluate their performance under the worst of the conditions. On the other hand, *Pypingen* was designed to begin its search from the "bottom" part of the search space and thus, it is also an evaluation in the worst case scenario. The conclusion that follows is that, compared with the state-of-the-art algorithms, *Pypingen* is able to explore a much larger region of the search space.

It is clear nonetheless that given a large enough dataset *Pypingen* will not be able to explore the entire search space and thus, it will be unable to mine frequent subsequences. Under these circumstances, frequent miners have the advantage of obtaining a set of results out of reach for a rare pattern miner such as *Pypingen*. In this light, we highlight the *benefits* of our approach. Like frequent miners use a threshold for support, we can provide a threshold for sequence size to mine *large sequences*. Using a proper value for this threshold, we can obtain all sequences for this restriction and particularly, those that are most frequent. Mining this type of *frequent large sequences* is not possible for frequent miners. Another threshold

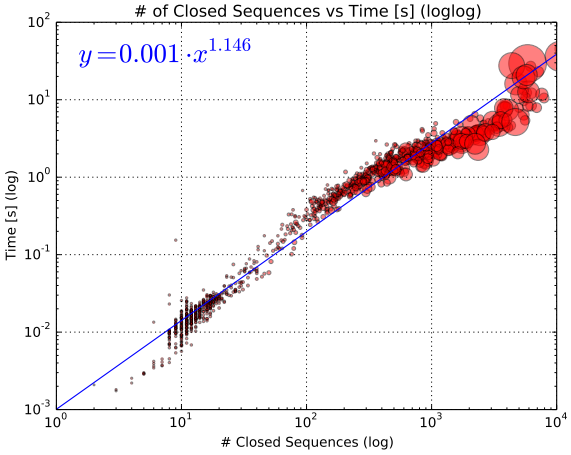



**Fig. 5.** Number of closed subsequences vs execution time for 1200 datasets.

that *Pypingen* can easily support is the minimum cardinality per itemset of a sequence pattern, adding in the capabilities of restricting the complexity of the desired patterns.

Finally, one of the main features of *Pypingen* is its flexibility. Consider the intersection table that we use to generate the DAG of alignments such as the one shown in Table 3. In here, the intersection operation may be easily replaced by an abstract similarity operator that holds under idempotence, commutativity and associativity to support sequences of elements other than itemsets, in the same way that it is done when defining pattern structures. Such adaptation would allow supporting sequences of complex elements such as intervals, taxonomical elements or heterogeneous representations [8]. This kind of sequences of complex elements would be hard to mine in current sequence mining algorithms designed for prefix enumeration.

## 7 Conclusions

In this article we have introduced the main notions behind *rare sequence pattern* mining using the framework of formal concept analysis, particularly the "pattern structures" extension to deal with complex object descriptions. Indeed, we have modelled patterns as sequence sets where the similarity operator allows formalizing an algorithm for extracting closed subsequences.

We have shown how the similarity operator can be implemented using a directed acyclic graph (DAG) of sequence alignments where a set of sequences can be easily encoded. The DAG of alignments also allows for an easy interpretation of sequence patterns as well as for pruning procedures that allows cutting down the search space.

We have implemented our approach in a system called *Pypingen* and showed that w.r.t. state-of-the-art closed sequence mining algorithms, it performs better in the worst of the cases, this is, when obtaining the full set of closed subsequences from a database.

Finally, we have discussed the implications of our system and its possible extensions regarding the mining of sequences composed by elements other than itemsets.

## References


1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering. pp. 3–14. ICDE '95, IEEE Computer Society, Washington, DC, USA (1995)
2. Ayouni, S., Laurent, A., Yahia, S.B., Poncelet, P.: Mining closed gradual patterns. In: Artificial Intelligence and Soft Computing, 10th International Conference, ICAISC 2010, Zakopane, Poland, June 13-17, 2010, Part I. Lecture Notes in Computer Science, vol. 6113, pp. 267–274. Springer (2010)
3. Buzmakov, A., Egho, E., Jay, N., Kuznetsov, S.O., Napoli, A., Raïssi, C.: On Projections of Sequential Pattern Structures (with an application on care trajectories).

In: Ojeda-Aciego, M., Outrata, J. (eds.) The Tenth International Conference on Concept Lattices and their Applications - CLA 2013, La Rochelle, France. pp. 199–208. Université de La Rochelle (2013)
4. Casas-Garriga, G.: Summarizing sequential data with closed partial orders. In: Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005. pp. 380–391. SIAM (2005)
5. Fabrègue, M., Braud, A., Bringay, S., Le Ber, F., Teisseire, M.: Mining closed partially ordered patterns, a new optimized algorithm. Knowledge-Based Systems 79, 68–79 (2015)
6. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: Conceptual Structures: Broadening the Base, 9th International Conference on Conceptual Structures, ICCS 2001 (2001)
7. Gomariz, A., Campos, M., Marin, R., Goethals, B.: ClaSP: An Efficient Algorithm for Mining Frequent Closed Sequences. In: Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013 (2013)
8. Kaytoue, M., Codocedo, V., Buzmakov, A., Baixeries, J., Kuznetsov, S.O., Napoli, A.: Pattern structures and concept lattices for data mining and knowledge processing. In: ECML PKDD 2015. pp. 227–231 (2015)
9. Mooney, C.H., Roddick, J.F.: Sequential pattern mining – approaches and algorithms. ACM Computing Surveys 45(2), 19:1–19:39 (2013)
10. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q.C.Q., Dayal, U., Hsu, M.C.H.M.C.: PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. Proceedings 17th International Conference on Data Engineering pp. 215–224 (2001)
11. Wang, J., Han, J.: BIDE: efficient mining of frequent closed sequences. In: Data Engineering, 2004. Proceedings. 20th International Conference on. pp. 79–90 (2004)
12. Yan, X., Han, J., Afshar, R.: Clospan: Mining: Closed sequential patterns in large datasets. In: Proceedings of the 2003 SIAM International Conference on Data Mining. pp. 166–177 (2003)
13. Zaki, M.J.: SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning 42(1-2), 31–60 (2001)
14. Zhang, J., Wang, Y., Yang, D.: CCSpan: Mining closed contiguous sequential patterns. Knowledge-Based Systems 89, 1–13 (2015)